Rapport du projet de statistique bayésienne

A partir de l'article *Variable selection in clustering via Dirichlet process mixture models*
S. Kim, M.G. Tadesse M. Vannucci

Christophe MORAU, Emilien JEMELEN

1 Présentation du problème et de son intérêt

Les méthodes de statistiques en haute dimension permettent d'effectuer de l'inférence sur des jeux de données contenant davantage de colonnes que d'observations et ont donc gagné en importances ces dernières années. Elles nécessitent généralement de supposer la sparsité du modèle : le nombre de variables véritablement pertinentes est (significativement) plus petit que le nombre de variables intégrées dans le jeu de données.

Appliquées au clustering, ces méthodes doivent permettre de dégager la structure intrinsèque de jeux de données qui n'ont pas été préalablement labellisés, ce qui est intéressant pour tous types de données possédant beaucoup de variables et en particulier dans le champ médical : appliquées à l'ADN, ces méthodes peuvent par exemple permettre de trouver les gènes différenciant des patients atteints d'une même maladie, et ainsi de trouver des traitements pertinents pour chacun des sous-groupes de patients.

Si des méthodes fréquentistes ou bayésiennes efficaces ont été développées en régression ou en classification, le problème est davantage complexe en non-supervisé. L'absence de labels rend notamment l'apprentissage de la sparsité difficile. Les contributions antérieures à celle de Kim, Tadesse et Vannucci sur le problème de clustering en haute dimension étaient ainsi rares et ne permettaient pas en général d'inférer le véritable nombre de clusters.

L'ambition de ce nouvel article était alors, en s'appuyant sur les précédents travaux d'un des auteurs, de proposer une méthode permettant d'inférer simultanément la structure de sparsité et l'allocation des clusters. Pour cela, une approche bayésienne a été développée afin d'estimer à la fois une variable latente décrivant la structure de sparsité et l'allocation des clusters à partir de modèles de mélanges utilisant les processus de Dirichlet.

2 Méthodologie

Soit $X = (x_1, ...x_n)$ un vecteur d'observations indépendantes, chacunes de dimension p avec $p << n$. On suppose que ces observations peuvent être séparées en différents sous-groupes homogènes (ou clusters). On fait l'hypothèse de sparsité : seul un sous ensemble d'indices $\gamma \subset [\![1, p]\!]$ avec $|\gamma| << p$ est tel que $\forall i \in [\![1, n]\!], (x_i)_\gamma$ contient les éléments discriminants des observations permettant de construire les clusters. Par la suite, on notera en fait γ comme un vecteur de dimension p dont la composante i vaut 1 si l'indice i est discriminant et 0 sinon. L'ambition de l'article est alors double : pouvoir identifier simultanément γ et les clusters.

2.1 Obtention de la posteriori des clusters sachant γ

Etant donné un ensemble γ, l'article fait l'hypothèse que les éléments discriminants d'un cluster ont été générés par une même loi : $x_{i(\gamma)} \sim F(\theta_i)$ et si $\theta_i = \theta_l$ alors les observations i et l appartiennent au même cluster. Les paramètres θ_i sont modélisés comme des variables aléatoires de loi G. Cette loi G étant inconnue, on adoptera une approche bayésienne en posant une priori dessus. Etant donné qu'il s'agit d'une priori sur une loi de probabilité, les auteurs proposent d'utiliser comme priori un processus de Dirichlet. On peut alors voir le modèle sur les θ_i comme un modèle de mélange au nombre de composants inconnu (le nombre de clusters, c'est à dire le nombre de θ_i uniques).

Un processus de Dirichlet peut également être vu comme une extension quand $K \to \infty$ de la loi de Dirichlet, qui est une priori pertinente dans le cas d'un modèle de mélange au nombre de composants K connu. Cela amène les auteurs à réécrire le modèle comme le cas limite quand $K \to \infty$ du modèle suivant :

$$\begin{aligned} x_{i(\gamma)}|c_i, \phi &\sim F(\phi_{c_i}) \\ c_i|p &\sim Discrete(p_1, ..., p_K) \\ \phi_c &\sim G_0 \\ p &\sim Dir(\alpha/K, ..., \alpha/K) \end{aligned}$$

avec G_0 l'espérance du processus de Dirichlet, α son paramètre d'échelle, c_i le cluster alloué à l'observation i et ϕ_{c_i} le paramètre caractérisant la loi des éléments discriminants des observations appartenant à ce cluster. Alors le modèle de mélange sur les clusters s'écrit :

$$c_i|c_{-i}, p \sim \sum_{k\in[\![1,K]\!],k\neq i} \frac{1}{n-1+\alpha} p_k \quad + \frac{\alpha}{n-1+\alpha} G_0$$

En intégrant p et en prenant la limite $K \to \infty$, on obtient finalement la priori suivant sur les clusters :

$$pr(c_i = c_l \text{ pour un } l \neq i|c_{-i}) = \frac{n_{-i,c_l}}{n-1+\alpha}$$
$$pr(c_i \neq c_l \text{ pour tout } l \neq i|c_{-i}) = \frac{\alpha}{n-1+\alpha}$$

avec n_{-i,c_l} le nombre d'observations hormis l'observation i qui sont allouées au cluster c_l. En choisissant bien G_0 de façon à ce qu'elle soit conjuguée avec F, on a alors, toujours pour γ connu, la posteriori suivante :

$$pr(c_i = c_l \text{ pour un } l \neq i|c_{-i}, x_i, \gamma) = b\frac{n_{-i,c_l}}{n-1+\alpha} \int F(x_{i(\gamma)}, \phi) dH_{-i,c_l}(\phi)$$
$$pr(c_i \neq c_l \text{ pour tout } l \neq i|c_{-i}, x_i, \gamma) = b\frac{\alpha}{n-1+\alpha} \int F(x_{i(\gamma)}, \phi) dG_0(\phi)$$

avec b une constante de normalisation et où l'on a intégré ϕ (selon G_0 lorsque l'on engendre un nouveau paramètre ϕ_{c_k}, selon H_{-i,c_l} soit la posteriori de ϕ_{c_l} lorsque l'on adopte la valeur c_l).

En pratique, on utilisera ce résultat en supposant que F correspond à une loi normale : $x_{i(\gamma)}|c_i = k, \phi_k, \gamma \sim \mathcal{N}(\mu_{k(\gamma)}, \Sigma_{k(\gamma)})$. On a alors $\phi_k = (\mu_{k(\gamma)}, \Sigma_{k(\gamma)})$ et on pose également des prioris sur ces paramètres : $\mu_{k(\gamma)}|\Sigma_{k(\gamma)}) \sim \mathcal{N}(\mu_{0(\gamma)}, h_1\Sigma_{k(\gamma)})$ et $\Sigma_{k(\gamma)} \sim IW(\delta, Q_{1(\gamma)})$

2.2 Obtention de la posteriori de γ sachant c

γ_i vaut 1 si l'indice i est discriminant et 0 sinon. Cela nous amène à proposer comme priori que le vecteur γ est composé de Bernoullis indépendantes de même paramètre ω : $pr(\gamma) = \prod_{j=1}^{p} \omega^{\gamma_j}(1-\omega)^{1-\gamma_j}$. Afin d'obtenir la posteriori de $\gamma|c$, nous avons besoin de connaitre la vraisemblance $X|\gamma, c$. On a déjà vu que l'on supposait $x_{i(\gamma)}|c_i = k, \phi_k, \gamma \sim \mathcal{N}(\mu_{k(\gamma)}, \Sigma_{k(\gamma)})$. Si l'on suppose l'absence de corrélations entre les éléments discriminants et non discriminants, il sera suffisant d'également spécifier la distribution de $x_{i(\gamma^c)}|c, \gamma$. Les éléments de γ^c sont non discriminants, on suppose alors la même loi sur toutes les observations et on suppose à nouveau qu'elles suivent une loi gaussienne : $x_{i(\gamma^c)}|\gamma, \eta, \Omega \sim \mathcal{N}(\eta_{\gamma^c}, \Omega_{\gamma^c})$. On prend comme priori pour η et Ω des lois gaussiennes et inverse gamma : $\eta_{\gamma^c}|\Omega_{\gamma^c} \sim \mathcal{N}(\mu_{0(\gamma^c)}, h_0\Omega_{\gamma^c})$ et $\Omega = \sigma^2 I_p$ avec $\sigma^2 \sim IG(a, b)$

En notant $\psi = (\eta, \Omega)$, la vraisemblance $X|c, \gamma, \phi, \psi$ est le produit des vraisemblances $X_{(\gamma^c)}|\gamma, \psi$ et $X_{(\gamma)}|c, \phi, \gamma$.

En intégrant ψ et ϕ, on obtient finalement :

$$f(X|\gamma, c) = \pi^{-np/2} H_{0(\gamma^c)} S_{0(\gamma^c)}^{-(a+n/2)} \prod_{k=1}^{K} H_{k(\gamma)} |Q_{1(\gamma)}|^{(\delta+p_\gamma-1)/2} |Q_{1(\gamma)} + S_{k(\gamma)}|^{-(n_k+\delta+p_\gamma-1)/2}$$

avec

$$H_{k(\gamma)} = (h_1 n_k + 1)^{-p_\gamma/2} \prod_{j=1}^{p_\gamma} \frac{\Gamma((n_k+\delta+p_\gamma-j)/2)}{\Gamma((\delta+p_\gamma-j)/2)}$$

$$H_{0(\gamma)} = (h_0 n + 1)^{-(p-p_\gamma)/2} b^{a(p-p_\gamma)} \prod_{j=1}^{p-p_\gamma} \frac{\Gamma(a+n/2)}{\Gamma(a)}$$

$$S_{k(\gamma)} = \sum_{i\in C_k} (x_{i(\gamma)} - \overline{x}_{k(\gamma)})(x_{i(\gamma)} - \overline{x}_{k(\gamma)})^T + \frac{n_k}{h_1 n_k + 1} (\mu_{0(\gamma)} - \overline{x}_{k(\gamma)})(\mu_{0(\gamma)} - \overline{x}_{k(\gamma)})^T$$

$$S_{0(\gamma^c)} = \prod_{j=1}^{p-p_\gamma} [b + \frac{1}{2}(\sum_{i=1}^{n} (x_{ij(\gamma^c)} - \overline{x}_{j(\gamma^c)})^2 + \frac{n}{h_0 n + 1} (\mu_{0j(\gamma^c)} - \overline{x}_{j(\gamma^c)})^2)]$$

avec $\overline{x}_{k(\gamma)}$ la moyenne des échantillons sur le cluster k et $\overline{x}_{j(\gamma^c)}$ la moyenne des échantillons sur la j-eme variable non discriminante.

La posteriori correspond alors à $f(\gamma|X, c) \propto f(X|\gamma, c) pr(\gamma)$

3 Mise en place de l'algorithme décrit dans l'article

L'algorithme qui permet de faire évoluer conjointement γ et c repose sur la procédure introduite par Jain et Neal's (2004). Pour chaque itération de l'algorithme (dont le nombre total est fixé par l'utilisateur) :

γ est modifié comme suit : pour i allant de 1 à t (t $\leq$ 20 recommandé dans l'article)

- Avec probabilité 0.5, l'un des deux changements suivants est exploré dans une copie de γ appelée γ_{new} :
  → Changement 1 : l'un des éléments de γ choisi aléatoirement voit sa valeur changée (0 si 1, 1 si 0)
  → Changement 2 : un élément dont la valeur est 0 est tiré aléatoirement, un élément dont la valeur est 1 est tiré aléatoirement, leurs valeurs respectives sont permutées
- Le vecteur latent d'exploration γ_{new} est adopté avec probabilité $min\{1, \frac{f(\gamma_{\text{new}}|X,c)}{f(\gamma|X,c)}\} \underset{Bayes}{=} min\{1, \frac{f(X|\gamma_{\text{new}},c).\pi(\gamma_{\text{new}})}{f(X|\gamma,c).\pi(\gamma)}\}$ en comparant cette quantité avec un tirage u d'une loi uniforme sur $[0,1]$ (γ_{new} accepté si u est plus petit)

c est ensuite modifié selon une procédure dite de *split-merge* ("séparation fusion") détaillée par Jain et Neal (2004) : on répète les étapes ci-après jusqu'à ce que toutes les composantes de c aient encouru une modification

Deux éléments distincts i et j de c, à valeurs c_i et c_j, sont tirés aléatoirement

On considère $\mathcal{C}$ l'ensemble des éléments de c dont la valeur est égale à c_i ou c_j

→ Si $\mathcal{C}$ est vide, deux cas se présentent :

Cas 1 : $c_i = c_j$: le *split* consiste alors à donner à l'élément i une nouvelle valeur qui n'est pas encore attribuée dans c. On obtient un vecteur d'exploration c_{split}, accepté avec probabilité $min\{1, \frac{\alpha L(c_{split|X,\gamma})}{L(c|X,\gamma)}\}$. **Note** : *dans Neal (2000), l'élément i reçoit la valeur d'un cluster existant avec une probabilité proportionnelle à la taille du cluster, et reçoit une valeur nouvelle inexistante dans c avec une probabilité proportionnelle à α. Dans l'article, le paramètre α est fixé arbitrairement à 1.*
Cas 2 : $c_i \neq c_j$: le *merge* consiste alors à attribuer à c_i la valeur c_j. On obtient un vecteur d'exploration c_{merge}, accepté avec probabilité $min\{1, \frac{L(c_{merge|X,\gamma})}{\alpha L(c|X,\gamma)}\}$

→ Si $\mathcal{C}$ est non vide, une procédure d'échantillonnage de Gibbs est proposée par l'article :

Construction d'un "état de lancement" c_{launch} : c_i et c_j font l'objet d'un *split* ou d'un *merge* en fonction de leurs valeurs (comme dans le cas où $\mathcal{C}$ est vide), puis chacun des éléments de c_{launch} indexés dans $\mathcal{C}$ se voit attribué la valeur c_{launch_i} ou bien c_{launch_j} (avec probabilité 0.5). Des itérations d'une procédure de Gibbs sampling restreinte utilisant les lois conditionnelles $c_i|c_{-i}, x_i, \gamma$ et $c_l|c_{-l}, x_i, \gamma$ fixent l'état final de cet état de lancement. Ce Gibbs sampling est dit restreint dans la mesure où il ne permet de prendre que deux valeurs, ici c_i ou c_l. Les auteurs annoncent que les changements sont minimes après plus de 5 itérations.
Echantillonnage de Gibbs final : à partir de c_{launch}, c_{split} est construit si $c_i = c_j$ en assignant aux éléments de c_{launch} indexés par $\mathcal{C}$ la valeur c_{launch_i} ou bien c_{launch_j} en utilisant une dernière itération de la procédure de Gibbs sampling restreinte. Les autres éléments restent inchangés. c_{split} est accepté avec probabilité :

$$min\{1, \frac{L(c_{split}|X,\gamma).pr(c_{split}).q(c|c_{split})}{L(c|X,\gamma).pr(c).q(c_{split}|c)}\}$$

Si $c_i \neq c_j$, c_{merge} est construit en assignant aux éléments de c_{launch} indexés par $\mathcal{C}$ la valeur c_j. Les autres éléments restent inchangés. c_{merge} est accepté avec probabilité :

$$min\{1, \frac{L(c_{merge}|X,\gamma).pr(c_{merge}).q(c|c_{merge})}{L(c|X,\gamma).pr(c).q(c_{merge}|c)}\}$$

→Enfin, un derniere itération de la procédure de Gibbs sampling, complète cette fois, est réalisée en utilisant les probabilités $c_i = c_l$ *pour un* $l \neq i | c_{-i}, x_i, \gamma$ et $c_i \neq c_l$ *pour tout* $l \neq i | c_{-i}, x_i, \gamma$

Remarques :

- La probabilité d'acceptation d'une proposition dépend du paramètre d'échelle α, qui, s'il est augmenté, augmente la probabilité d'acceptation des propositions de *split*. De même, la probabilité de l'événement $c_i \neq c_l$ *pour tout* $l \neq i | c_{-i}, x_i, \gamma$ utilisé dans le Gibbs samplign final est proportionnelle à α. Avoir α faible permet de réduire l'impact de la priori posé avec le processus de Dirichlet puisqu'on sera moins souvent amené à utiliser G_0, mais cela rend également généralement la probabilité de cet événement négligeable devant celle de l'événement $c_i = c_l$ *pour un* $l \neq i | c_{-i}, x_i, \gamma$.

  Pour ces deux raisons, la chaîne telle que nous l'avons codée selon l'article (avec $\alpha = 1$) va avoir tendance à privilégier les merges aux splits et à remplacer c_i par un autre c_l lors du dernier Gibbs sampling. De ce fait on va avoir tendance à se trouver bloqué dans un état mono-cluster de c.

- Les auteurs évoquent succintement la possibilité que la chaîne soit coincée en c sur un état sans accepter aucun des états d'exploration *split merge* proposés par l'algorithme. Ils évoquent aussi une méthode d'interpolation des distributions de c définie par Geyer (1991), qui permettrait de débloquer la chaîne.

- Concernant le fait que la chaîne se bloque en position mono-cluster pour c, nous n'avons pas réussi à savoir si le problème était lié au code lui-même ou bien au temps requis pour faire tourner l'algorithme. En effet, les auteurs exposent des résultats de convergence au bout de plusieurs dizaines de milliers d'itérations de la chaîne, soit plusieurs semaines avec notre puissance de calcul en local. Dans le temps imparti, il nous a été impossible de conduire des tests de cet ordre de grandeur.

  Nous avons tout de même entrepris de recalculer à la main les formules utilisées dans l'algorithme avec des exemples simples afin de vérifier que notre code donnait les mêmes résultats : cela nous a permis d'identifier certains problèmes mineurs dans notre implémentation mais rien de suffisant pour changer les résultats obtenus.

- Certains points de notre code font appel à notre interprétation personnelle d'éléments qui n'étaient pas explicites. Notamment, la définition des probabilité de transition q dans la probabilité d'acceptation : nous avons pu nous référer à l'article de Jain et Neal (2004) pour mieux comprendre ce point mais nous ne sommes pas certains que notre implémentation de ces probabilités soit correcte.

4 Mise en place d'une alternative fonctionnelle

Devant les difficultés rencontrées pour débloquer l'étape d'échantillonnage de Gibbs de l'algorithme d'origine décrit dans l'article, nous avons pris le parti de mettre en place une version alternative de cette étape de l'algorithme d'origine.
Rien ne change pour ce qui est de la mise à jour du vecteur latent γ contenant des 1 là où les colonnes sont discriminantes. En revanche, la mise à jour du vecteur des clusters c se fait dans notre alternative via un nombre fixe de modifications stochastiques soumises à une probabilité d'acceptation. Concrètement, après avoir mis à jour γ selon la méthodologie de l'article, c est mis à jour de la manière décrite ci-dessous.

4.1 Algorithme alternatif de mise à jour de c

Pour i allant de 1 à K :

→ Tirer aléatoirement deux samples
→ Si leurs labels dans c sont identiques, l'un des deux samples se voit attribué un label qui n'est pas encore présent dans c (ce que l'article appelle un *split*)
→ Le *split* est accepté avec probabilité

$$min\{1, \frac{L(c^{split}|X,\gamma).q(c|c^{split}).pr(c^{split})}{L(c|X,\gamma).q(c^{split}|c).pr(c)}\}$$

où, comme à l'étape 2 décrite dans l'article, on approxime

$$\frac{q(c|c^{split}).pr(c^{split})}{q(c^{split}|c).pr(c)} = \alpha$$

→ Si leurs labels dans c sont différents, l'un des deux samples se voit attribué le label de l'autre sample (ce que l'article appelle un *merge*)

→ Le *merge* est accepté avec probabilité

$$min\{1, \frac{L(c^{merge}|X,\gamma).q(c|c^{merge}).pr(c^{merge})}{L(c|X,\gamma).q(c^{merge}|c).pr(c)}\}$$

où, comme à l'étape 2 décrite dans l'article, on approxime

$$\frac{q(c|c^{merge}).pr(c^{merge})}{q(c^{merge}|c).pr(c)} = \frac{1}{\alpha}$$

Il s'agit en fait simplement d'un nombre K d'itérations de l'algorithme de Metropolis-Hastings. On saisit l'importance des paramètres K et α dans la mise à jour de c.

4.2 Méthode adoptée pour choisir les valeurs des paramètres K et α :

K : empiriquement, le nombre d'itérations nécessaire pour qu'en moyenne un changement de label au moins ait lieu pour chaque sample (sur des données simulées de taille 15 samples, K = 200 semble suffire).

Quant à α, l'idée est de le choisir de sorte qu'on ait des probabilités d'acceptation de *split* et de *merge* vérifiant un taux d'exploration suffisant, afin d'éviter d'avoir la chaîne bloquée dans un état mono-cluster de c.

En se basant sur l'article Roberts, Gelman et Gilks (1997) selon lequel la convergence des algorithmes de marche aléatoire de Metropolis-Hastings est optimale lorsque la probabilité d'acceptation vaut 0.234, on cherche, en approximant la marche aléatoire par le fait de tirer un *split* de c, α tel que la probabilité d'acceptation d'un *split* soit environ égale à cette valeur.

Algorithme de calibration de α

→ On lance trois itérations de Metropolis-Hastings pour γ et pour c (avec K = 200) en prenant $\alpha = 1$
→ On considère $V_{\mathbb{P}_{split}}$ le vecteur contenant les probabilités d'acceptation des *split* de c sauf celles valant 1 stockées au fil des trois itérations de la chaîne
→ On cherche α^* tel que la médiane du vecteur $V_{\mathbb{P}_{split}}$ soit environ égale à 0.234 en résolvant $\alpha^*.median(V_{\mathbb{P}_{split}}) = 0.234$ **(E)**
→ $\alpha = \alpha^*$

NB : on résout l'équation **(E)** car on suppose que le *split* est accepté avec probabilité

$$min\{1, \alpha.\frac{L(c^{split}|X,\gamma)}{L(c|X,\gamma)}\}$$

Quand $\alpha = 1$, $V_{\mathbb{P}_{split}}$ contient alors les rapports des vraisemblances pour les différentes propositions de *split*.

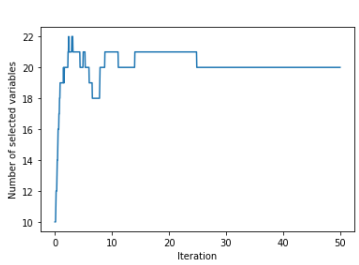
4.3 Tests de l'algorithme alternatif sur des données simples

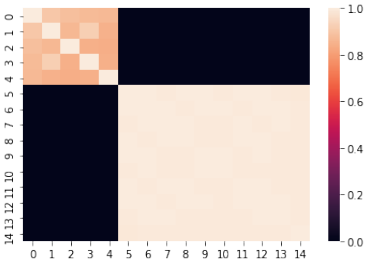
Une fois que la chaîne a fait sa dernière itération, n'ayant pas accès aux probabilités conditionnelles avec cette manière de mettre à jour c, on procède à l'extraction du vecteur c final avec une méthode alternative. On procède via une méthode connue de co-occurrence d'objets appelée matrice de co-association, dont on trouve le détail en annexes de ce rapport.

2 scenarii sont considérés (avec un nombre de samples fixé à 15):

Scenario	Nombre d'itérations	Nombre de variables discriminantes	Paramètres de la loi de simulation des variables discriminantes	Paramètres de la loi de simulation des variables non discriminantes
2 clusters - 100 variables	50	20	Cluster 1 = normale(-50, 0.04, size : (5, 20)) Cluster 2 = normale(300, 0.7, size : (10, 20))	normale(0, 1, (15, 80))
2 clusters - 200 variables	100	20	Cluster 1 = normale(-50, 0.04, size : (5, 20)) Cluster 2 = normale(300, 0.7, size : (10, 20))	normale(0, 1, (15, 180))

Résultats pour 2 clusters - 100 variables :



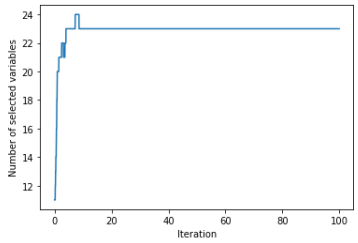


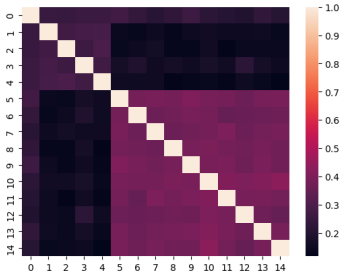
Gauche : Evolution du nombre de variables sélectionnées dans γ
Droite : Matrice de co-association du scenario 2 clusters - 100 variables

Interprétation : on voit immédiatement deux clusters apparaître, avec des coefficients d'association très éloignés (1 ou 0), et l'algorithme retrouve exactement (via la matrice de co-association) les vrais labels des données simulées

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
clust labels	1	1	1	1	1	2	2	2	2	2	2	2	2	2	2
true labels	1	1	1	1	1	2	2	2	2	2	2	2	2	2	2

Résultats pour 2 clusters - 200 variables :





Gauche : Evolution du nombre de variables sélectionnées dans γ
Droite : Matrice de co-association du scenario 2 clusters - 200 variables

Interprétation : on voit immédiatement deux clusters apparaître, mais les coefficients d'association sont beaucoup plus resserrés et aussi beaucoup plus bas, ce qui fait perdre de la robustesse au résultat. On retrouve malgré tout un résultat très proche des vrais labels en prenant un seuil d'association des points suffisamment faible :

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
clust labels	1	2	2	2	2	1	1	1	1	1	1	1	1	1	1
true labels	1	1	1	1	1	2	2	2	2	2	2	2	2	2	2

4.4 Limites de l'alternative et conclusion

L'alternative est moins rigoureuse puisqu'on se contente du rapport de vraisemblances pour adopter ou rejeter les états exploratoires de c. Toutefois, l'alternative semble plus rapide, et fonctionne pour des données simples mais tout de même réalistes : il s'agit déjà, avec 15 samples et 100 voire 200 variables, de haute dimension.

Pour conclure, nous aurions aimé mettre en place la méthode exacte de l'article, mais, ne parvenant pas à débloquer le problème de la chaîne qui se bloque en position mono-cluster en c, et, après plusieurs reprises minutieuses des formules, ne sachant pas si en réalité le problème se débloquerait après plusieurs semaines d'attente (de surcroît sous réserve que nos machines à la puissance limitée tiennent bon), nous avons mis en place une version légèrement modifiée de l'algorithme original, avec un paramètre α dont nous avons cherché à automatiser l'optimisation.

Bibliographie

G.O. Roberts, A. Gelman, W.R. Gilks, Weak convergence and optimal scaling of random walk Metropolis algorithms, Ann. Appl. Probab. 7 (1997) 110–120

Kim, Sinae, et al. "Variable Selection in Clustering via Dirichlet Process Mixture Models." Biometrika, vol. 93, no. 4, 2006, pp. 877–93. JSTOR, http://www.jstor.org/stable/20441333. Accessed 30 Dec. 2022.

Jain, S. Neal, R. M. (2004). A split-merge Markov chain Monte Carlo procedure for the Dirichlet process mixture model. J. Comp. Graph. Statist. 13, 158–82

ANNEXES

§ Compléments sur le consensus des partitions via la matrice de co-association :

Les approches de consensus par co-association s'attachent à représenter les résultats de la génération de nombreuses partitions d'un même jeu de données par un objet intermédiaire : la matrice de co-association, notée CA, définie de la manière (n samples, m partitions générées) : $\forall i, j \in 1, ..., n$,

$$CA_{i,j} := \frac{1}{m} \sum_{k=1}^{m} \mathbb{1}(P_k(x_i) = P_k(x_j))$$

Où $P_k(x_i)$ est le label du sample n°i dans la partition générée n°k.

$CA_{i,j}$ est une mesure dans $[0, 1]$ de la fréquence du nombre de fois où les samples i et j se trouvent classés dans le même cluster sur l'ensemble des partitions générées. En règle générale, il est décidé que deux points doivent se retrouver dans le même cluster si leur coefficient de co-association est supérieur ou égal à 0,5. Toutefois, il est possible de choisir un seuil d'association plus petit ou plus grand. Plus petit, le seuil est moins significatif et la partition finale de consensus obtenue par la matrice de co-association contiendra moins de clusters : un seuil de 0 fait que tous les points sont associés, donc la partition finale sera mono-cluster. Plus grand, le seuil est plus significatif et la partition finale de consensus obtenue par la matrice de co-association contiendra plus de clusters : un seuil de 1 fait généralement que chaque point est dans son propre cluster.